

Lisa

Getopt()

# Programmi käsureavalikud

- ♦ Kokkuleppeliselt tähistatakse programmi valikud käsureal miinusmärgiga (-) ja ühe tähega (ei ole päris tõsi, aga arvake praegu et on)
- ♦ Valikuid võib üksteise otsa kirjutada (kui neil pole parameetreid)
- ♦ Valikutel võivad olla parameetreid

```
myprog -a -b
```

```
myprog -ab
```

```
myprog -ba
```

```
myprog2 -i infile -ooutfile
```

- ♦ Kuidas sellises situatsioonis laisk olla?

# getopt()

```
#include <unistd.h>
int opterr;
int optopt;
int optind;
char * optarg;
int getopt(int argc, char**argv, const char *options)
```

- ♦ Tagastab valikute nimekirjast järgmise
- ♦ options näitab ära võimalikud "lubatud" käsureaavalikud, täht võimaliku tähemärgi, sellele järgnev koolon (:), et järgneb argument, ja (::) kui argument pole kohustuslik
- ♦ opterr: kui määrata 0, ei näita vigu välja
- ♦ optind: järgmine valik

## getopt() (2)

- ♦ getopt() vahetab tavaolukorras argv sees olevate valikute järjekorda kuni kõik miinuseta on lõpus
- ♦ Kui rohkem valikuid pole, tagastub -1
- ♦ Valikute lõppemisel saab kontrollida üle jäänud argumente võrreldes argc-d ja optind muutujat.
- ♦ Tundmatu valiku puhul tagastub '?', tundmatu märk salvestatakse muutujasse optopt

```
getopt(argc, argv, "abc:");
```

## getopt() (3)

- ♦ Reeglina kutsutakse välja tsükklis, mis lõpetab, kui getopt() tagastab -1
- ♦ Tagastatavat väärtust valitakse enamasti switch-lausega, kus väärtustatakse iga valiku puhul muutuja, mis hiljem programmi tööd vastavalt mõjutab
- ♦ Ülejäänud programmiargumentide jaoks kasutatakse teist tsüklit

# Näide

```
#include <unistd.h>
#include <stdio.h>

int
main (int argc, char **argv)
{
    int aflag = 0;
    int bflag = 0;
    char *cvalue = NULL;
    int index;
    int c;

    opterr = 0;

    while ((c = getopt (argc, argv, "abc::")) != -1)
```

## Näide (2)

```
switch (c) {
    case 'a':
        aflag = 1;
        break;
    case 'b':
        bflag = 1;
        break;
    case 'c':
        cvalue = optarg;
        break;
    case '?':
        if (isprint (optopt))
            fprintf (stderr, "Unknown option `-%c'.\n", optopt);
        else
            fprintf (stderr, "Unknown option char `\\x%x'.\n",
                    optopt);

        return 1;
    default:
        abort();
}
```

## Näide (3)

```
printf ("aflag = %d, bflag = %d, cvalue = %s\n",  
        aflag, bflag, cvalue);  
  
for (index = optind; index < argc; index++)  
    printf ("Non-option argument %s\n", argv[index]);  
return 0;  
}
```



# GNU täiendused

- ♦ GNU projekti raames on libc-s ka getopt\_long
- ♦ Selle eesmärgiks on programmide õppimine lihtsamaks teha:

```
ls -Ah  
ls --almost-all --human-readable
```

- ♦ Pikad valikud algavad topeltmiinusega (--)
- ♦ Pika valiku argumendid on kujul:  
--valikunimi=väärtus
- ♦ unistd.h asemel on prototüüp eraldi getopt.h failis

# struct option

- ♦ getopt\_long() võtab argumendiks massiivi valikuid kirjeldavatest structidest

```
struct option {  
    const char *name;  
    int has_arg;  
    int *flag;  
    int val;  
}
```

- ♦ name: valiku nimi
- ♦ has\_arg: no\_argument, required\_argument või optional\_argument
- ♦ flag ja val osas juurdleme põhjalikumalt kohe

## struct option (2)

- ♦ flag:
  - kui on pointer mingile väärtusele, salvestatakse **var** sisu näidatud aadressile, kui valik peaks ette juhtuma
  - kui on 0, on **var** mingi unikaalne number, mis valikut tuvastab (enamasti sama, mis oleks tavalise getopti puhul; pärast struct optioni näidet selgub, miks see hästi loogiline on)
- ♦ Massiivi viimane element peab olema struct option, mille kõik elemendid on 0

# struct option-i näide

```
static struct option long_options[] = {  
    /* siin on flag määratud. */  
    {"lamiseja", no_argument, &verbose_flag, 1},  
    {"tasane",    no_argument, &verbose_flag, 0},  
    /* siin on flag 0 .  
       eristamine käib tähtede/järjekorranr-i abil. */  
    {"add",      no_argument,      0, 'a'},  
    {"append",   no_argument,      0, 'b'},  
    {"delete",   required_argument, 0, 'd'},  
    {"create",   required_argument, 0, 'c'},  
    {"file",     required_argument, 0, 'f'},  
    {0, 0, 0, 0}  
};
```

# getopt\_long()

- Töötab lühikestega täpselt nagu tavaline getopt()
- Pikkadega sõltub valiku flag parameetrist: kas salvestab kuskile muutuja või tagastab väärtuse val
  - seega kui tagastame lühikese valiku tähe, saame kasutada ka pika puhul sama käitumist

```
int getopt_long (int argc, char *const *argv,  
                const char *shortopts,  
                const struct option *longopts,  
                int *indexptr)
```

- Kui ülaltoodust ei piisa, on olemas argp(), mis meenutab karujahti keskmaaraketiga: väga tõhus, aga sihtimine on veidi keeruline

# getopt\_long() näidis

```
c = getopt_long (argc, argv, "abc:d:f:",
                  long_options, &option_index);
if (c == -1) /* valikute ots */
    break;
switch (c) {
    case 0: /* kui lipp, ignoreerime praegu. */
        if (long_options[option_index].flag != 0)
            break;
        printf ("option %s",
                long_options[option_index].name);
        if (optarg)
            printf (" with arg %s", optarg);
        printf ("\n");
        break;
    case 'a':
        puts ("option -a\n");
        break;
    case 'b':
        /* ... */
}
```

# Getopt lõpetuseks

- ♦ `getopt()` on üsna lihtne, mugav ja veakindel
- ♦ `getopt_long()` on tunduvalt keerulisem, kuid on ikkagi mugavam, kui ise analoogne parsija kirjutada
- ♦ Kasutage!